Linear and Nonlinear Solvers in Truchas

John A. Turner

Telluride Workshop

(1/22-23/2003)

LA-UR-03-0318

Computer and Computational Sciences Division
Voice: (505) 667 - 4297
Fax: (505) 667 - 4972
turner@lanl.gov
www.ccs.lanl.gov

Solver needs in Truchas

Linear $\boxed{Ax = b}$

- flow
  - *pressure projection (Poisson)* $$\nabla \bullet \left(\frac{\delta t}{\rho} \nabla \delta p \right) = F(u)$$
  - *implicit viscous treatment* $$\left(\rho_c + c_d - \nabla \cdot \mu \nabla \right) u^* = F(u_c, u_f, P, \rho_c, \rho_f)$$
- thermo-mechanical $$G a_{i,jj} + (\lambda + G) a_{j,ji} - (3\lambda + 2G)\alpha (T - T_0)_i = 0$$
- E&M

Nonlinear $\boxed{F(x) = 0 \;\rightarrow\; J^k \delta x^k = -F(x^k)}$

- heat-transfer/phase change

Basics

linear system: $Ax = b, A \in \mathbb{R}^{mxn}, x \in \mathbb{R}^{n}, b \in \mathbb{R}^{m}$

residual and error: $r = b - Ax \qquad e = x - \hat{x}$

dot product: $x^T y = \sum_{i=1}^{n} x_i y_i$

norms: $\|x\|_2 = \sqrt{x^T x} \qquad \|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$

condition number: $\kappa(A) = \|A\| \|A^{-1}\|$

spectral radius: $\rho(A) = \max(|\lambda|) = \lambda_1$

orthonormal: $x^T y = 0, \|x\| = \|y\| = 1$

Linear Solver Approaches

- Linear Solver Types
  - Direct
    - Cholesky Factorization
    - LU Decomposition
  - Iterative
    - Classical Stationary
      - Jacobi
      - Gauss-Seidel
      - SOR
      - SSOR
    - Krylov Subspace
    - Multilevel
      - Grid-based (Structured)
      - Algebraic (Unstructured)
        - LAMG (LANL)
        - HYPRE (LLNL)

$$A = A_1 + A_2$$

$$x^{k+1} = Mx^k + c$$

$$M = -A_1^{-1} A_2$$

Stationary Iterative Methods

- convergence driven by $\rho(M) = \max(|\lambda_i|) = \lambda_1$
- almost always less efficient than Krylov or multilevel methods
- can be useful as preconditioners for Krylov methods

Krylov Subspace Methods

no iteration matrix

- instead, idea is to minimize some measure of error over the affine space $x_0 + \kappa_k$ where x_0 is the initial iterate and the k^{th} Krylov subspace is

$$\kappa_k = \mathrm{span}(r_0, Ar_0, \ldots, A^{k-1} r_0)$$

- original, and most well-understood, is method of conjugate gradients (CG)

Conjugate Gradients

- developed in '50's as a direct method
  - *in exact arithmetic solution guaranteed in n iterations*
- renewed interest in the '80's due to:
  - *realization that method could be viewed as iterative*
  - *preconditioning*
  - *pipelined vector computers*
- applicable to symmetric positive definite systems
- convergence governed by $\kappa_2(A) = \|A\|_2 \, \|A^{-1}\|_2 = \dfrac{\lambda_1(A)}{\lambda_n(A)}$ rather than by $\rho(A) = \max(|\lambda_i|) = \lambda_1$

COMPUTER & COMPUTATIONAL SCIENCES

Preconditioning

CG convergence improves if $A \approx I$

- recast $Ax = b$ as:

$$M^{-1}Ax = M^{-1}b \quad \text{or} \quad AM^{-1}y = b, x = M^{-1}y$$

(left preconditioning) (right preconditioning)

- $M^{-1}A \approx I$ implies that in some sense $M \approx A$
- must solve system of form $Mz = w$ at each iteration
- though general preconditioners available (Jacobi, SSOR, etc.), best preconditioners use knowledge of underlying physics/numerics

NNSA Los Alamos

CG Costs

most costly operations in CG algorithm are:

- matrix-vector multiplication (matvec)
- preconditioning (if used)

other operations needed:

- dot products (global communication)
- vector norms (global communication)
- other vector operations (addition, etc.)

Krylov Subspace Methods

- **Krylov Subspace**
  - **Hermitian Positive Definite**
    - Conjugate Gradients
  - **Hermitian Indefinite**
    - Conjugate Residuals — MINRES
    - Lanczos — SYMMLQ
  - **Non-Hermitian**
    - **Normal Equations**
      - CGNE — CGNR
      - LSQR
    - **Lanczos-based**
      - GCR — Bi-CG
      - CGS — Bi-CGstab
      - QMR — TFQMR
    - **Minimization**
      - Orthores — Orthodir
      - Orthomin — FOM
      - GCR — GMRES

give up minimization (Lanczos-based)

give up 3-term recurrence (Minimization)

Hermitian: $A = A^H$

Positive Definite:

$x^T A x > 0 \ \forall \ x > 0$

$\lambda_i > 0 \ \forall \ i$

GMRES

generally considered "best" method for non-Hermitian systems

- reduces to CG in symmetric case
- storage/computation increases each iteration
  - *extra vector of length n each iteration*
  - *larger triangular solve each iteration*
- mitigated by implementing as restarted algorithm

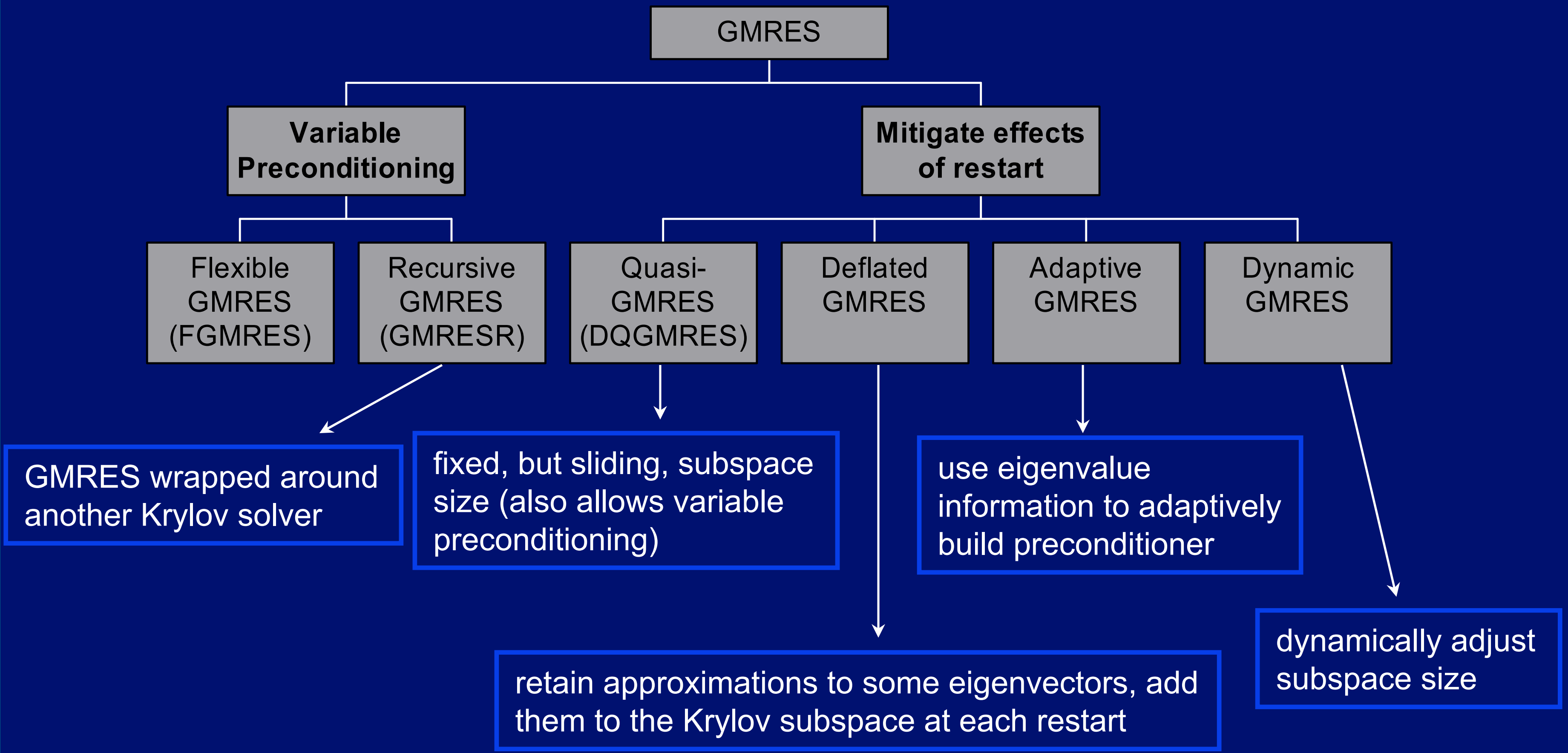
GMRES(k)

restarted GMRES

- every k iterations, set $x_0 = x_k$ and restart
- nice convergence properties lost
  - *convergence will be <u>at least</u> slowed*
  - *iterations can stagnate and fail to converge at all*
- some strategies to mitigate negative effects of restart have been developed
  - *best strategy, of course, is effective preconditioning*

LA-UR-03-0318

Extensions of GMRES



Truchas Solver Evolution: Overview

Early development:

- JTpack77 (F77 + extensions), then JTpack90 (F90)

Meta-stable mid-life:

- JTpack90 (most progress in Truchas rather than solver library)

Current development:

- Ubik (a descendent of JTpack90, F95)
  - *independent library available via SourceForge:*

http://sf.net/projects/ubiksolve/

Truchas Solver Issues

- unstructured meshes
- nonsymmetric operators
- coefficient matrices not explicitly formed (currently)
- parallel (scalable)
  - *domain decomposition*

Linear Solver options in Truchas

default solver is now FGMRES(k), but others available:

- CG
- GMRES(k)
- TFQMR
- Bi-CGstab

choice of Krylov subspace size (k) <u>critical</u> for GMRES(k) and FGMRES(k)

Stopping Tests

difficult to know when solution is "good enough"

- $\dfrac{\|r\|}{\|b\|}$ good unless $\|A\|\|x\| \gg \|b\|$
- $\dfrac{\|r\|}{\|A\|\|x\|+\|b\|}$ good, but we don't have $\|A\|$
- $\dfrac{\|r\|}{\|r_0\|}$ often used, but dependent on initial guess
- $\dfrac{\|r\|}{\|x\|}$ useful, but increases cost of GMRES, and not dimensionless
- $\|r\|$ not scaled, and not dimensionless, but often useful
- $\dfrac{\|x - x_{old}\|}{\|x\|}$ terrible for Krylov methods!

Preconditioners in Truchas

options available depends on the physics

all currently use an ortho approximation

- Jacobi, SSOR
- ILU(0) – Incomplete LU, no fill-in
- 2-Level Additive Schwarz

LA-UR-03-0318

Preconditioners in Truchas (cont.)

preconditioners and parallel

- global – Jacobi
  - *effectiveness independent of number of procs (nprocs), but requires communication*
- block Jacobi / additive Schwarz (1-level)
  - *Jacobi, SSOR, LU, ILU(0) for subdomain solves*
  - *nprocs* $\uparrow \longrightarrow$ *effectiveness* $\downarrow$
- 2-level additive Schwarz - SSOR

2-Level Additive Schwarz Preconditioning

construct single coarse grid

- perform subdomain solves on fine grid
- piecewise-constant restriction to coarse grid
- solve coarse grid system
- piecewise-constant prolongation and correction to fine grid
- subdomain solves on fine grid

2-Level Additive Schwarz Preconditioning (cont.)

usually quite effective

- much better scaling than block-Jacobi, since coarse-grid correction knocks out low-freq. error modes

flexible

- coarse grid corresponds to number of partitions, which doesn't necessarily have to equal number of processors

could be improved further

- better prolongation and restriction
- extend to multilevel

Newton-Krylov Algorithm

phase-change algorithm requires solution of nonlinear system

- Jacobian-free Newton-Krylov
  - *basically, Newton's method with a Krylov subspace method for the linear solves*

$$\boxed{F(x)=0 \;\rightarrow\; J^k\delta x^k=-F(x^k),\, x^{k+1}=x^k+\delta x^k}$$

Jacobian-Free N-K

Krylov subspace methods require Jacobian only for matrix-vector products

- approximate by first-order Taylor expansion

$$\boxed{\mathbf{J}\mathbf{y} \approx \frac{\left[\mathbf{F}(\mathbf{x}+\varepsilon\mathbf{y})-\mathbf{F}(\mathbf{x})\right]}{\varepsilon}}$$

  - *no need to form or invert actual Jacobian*
  - *perturbation factor:* $\varepsilon = \dfrac{1}{N\|\mathbf{y}\|_2}\displaystyle\sum_{i=1}^{N}\sqrt{u}\,|x_i|$

Detail on Jacobian-vector product approximation

consider two coupled nonlinear eqns.

$$F_1(x_1, x_2) = 0 \qquad F_2(x_1, x_2) = 0$$

$$\mathbf{Jy} = \begin{pmatrix} \dfrac{\partial F_1}{\partial x_1} & \dfrac{\partial F_1}{\partial x_2} \\ \dfrac{\partial F_2}{\partial x_1} & \dfrac{\partial F_2}{\partial x_2} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_1 \dfrac{\partial F_1}{\partial x_1} & y_2 \dfrac{\partial F_1}{\partial x_2} \\ y_1 \dfrac{\partial F_2}{\partial x_1} & y_2 \dfrac{\partial F_2}{\partial x_2} \end{pmatrix}$$

LA-UR-03-0318

Detail on Jacobian-vector product approximation (cont.)

$$\frac{\left[\mathbf{F}(\mathbf{x}+\varepsilon\mathbf{y})-\mathbf{F}(\mathbf{x})\right]}{\varepsilon}=\begin{pmatrix}\dfrac{F_1(x_1+\varepsilon y_1,x_2+\varepsilon y_2)-F_1(x_1,x_2)}{\varepsilon}\\[2ex]\dfrac{F_2(x_1+\varepsilon y_1,x_2+\varepsilon y_2)-F_2(x_1,x_2)}{\varepsilon}\end{pmatrix}$$

***first-order Taylor series expansion about* x**

$$\approx\begin{pmatrix}\dfrac{F_1(x_1,x_2)+\varepsilon y_1\dfrac{\partial F_1}{\partial x_1}+\varepsilon y_2\dfrac{\partial F_1}{\partial x_2}-F_1(x_1,x_2)}{\varepsilon}\\[3ex]\dfrac{F_2(x_1,x_2)+\varepsilon y_1\dfrac{\partial F_2}{\partial x_1}+\varepsilon y_2\dfrac{\partial F_2}{\partial x_2}-F_2(x_1,x_2)}{\varepsilon}\end{pmatrix}$$

Detail on Jacobian-vector product approximation (cont.)

$$\frac{\left[\mathbf{F}(\mathbf{x}+\varepsilon\mathbf{y})-\mathbf{F}(\mathbf{x})\right]}{\varepsilon}=\begin{pmatrix}\dfrac{F_1(x_1+\varepsilon y_1,x_2+\varepsilon y_2)-F_1(x_1,x_2)}{\varepsilon}\\[2ex]\dfrac{F_2(x_1+\varepsilon y_1,x_2+\varepsilon y_2)-F_2(x_1,x_2)}{\varepsilon}\end{pmatrix}$$

first-order Taylor series expansion about x

$$\approx\begin{pmatrix}\dfrac{\cancel{F_1(x_1,x_2)}+\cancel{\varepsilon}y_1\dfrac{\partial F_1}{\partial x_1}+\cancel{\varepsilon}y_2\dfrac{\partial F_1}{\partial x_2}-\cancel{F_1(x_1,x_2)}}{\cancel{\varepsilon}}\\[3ex]\dfrac{\cancel{F_2(x_1,x_2)}+\cancel{\varepsilon}y_1\dfrac{\partial F_2}{\partial x_1}+\cancel{\varepsilon}y_2\dfrac{\partial F_2}{\partial x_2}-\cancel{F_2(x_1,x_2)}}{\cancel{\varepsilon}}\end{pmatrix}=\begin{pmatrix}y_1\dfrac{\partial F_1}{\partial x_1} & y_2\dfrac{\partial F_1}{\partial x_2}\\[2ex] y_1\dfrac{\partial F_2}{\partial x_1} & y_2\dfrac{\partial F_2}{\partial x_2}\end{pmatrix}$$

LA-UR-03-0318

Recent improvements

FGMRES(k) as default linear solver

- right-preconditioning
- allows variable/adaptive preconditioning

improved output and diagnostics

- input variable to provide periodic status updates to `tty`
- on failure now clear which physics had problems
- on failure residuals dumped in GMV format

LA-UR-03-0318

Future Work

LAMG for preconditioner solves

- not expecting huge gains

form full least-squares operator explicitly

- Mike Hall recently showed that it's possible
- significant performance improvement expected, due to decreased MatVec cost
- allows use of LAMG on full operator

COMPUTER & COMPUTATIONAL SCIENCES

Future Work (cont.)

adaptive preconditioning

- may not work due to typical convergence behavior of Krylov methods

adaptive GMRES

- optimal situation is GMRES with a good enough preconditioner that restarting isn't required
- in reality restart will always be a possibility
- this could help mitigate problems inherent in restarting

NNSA Los Alamos

LA-UR-03-0318

Future Work (cont.)

nonlinear solver

- study interplay between convergence of both nonlinear and linear solves and timestepping

software issues

- componentization
  - *preconditioning*
  - *N-K*

LA-UR-03-0318